TUGAS KOMPUTASI PARAREL

Nama : Arman Surahman

Kelas : 7A Teknik Informatika

Mata Kuliah : Komputasi Pararel

Kode Program:

```c
#include <mpi.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define ROWS 100
#define COLS 100

int main(int argc, char** argv) {
    int rank, size;
    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);

    /* arrays for distribution */
    int *sendcounts = malloc(size * sizeof(int));     // jumlah elemen (ints) yang dikirim ke tiap proc
    int *displs_elems = malloc(size * sizeof(int));    // displacement dalam elemen (ints)
    int *rows_per_proc = malloc(size * sizeof(int));   // berapa baris tiap proc

    int base = ROWS / size;
    int rem  = ROWS % size;
    int offset_elems = 0;
    for (int i = 0; i < size; i++) {
        rows_per_proc[i] = base + (i < rem ? 1 : 0);
        sendcounts[i] = rows_per_proc[i] * COLS;   // setiap baris punya COLS elemen
        displs_elems[i] = offset_elems;
        offset_elems += sendcounts[i];
    }
```

```c
    /* root membuat matriks penuh */
    int *matrix = NULL;
    if (rank == 0) {
        matrix = malloc(ROWS * COLS * sizeof(int));
        srand(time(NULL)); // atau pake srand(42) supaya deterministic
        for (int r = 0; r < ROWS; r++) {
            for (int c = 0; c < COLS; c++) {
                matrix[r * COLS + c] = rand() % 100; // angka 0..99
            }
        }
    }

    /* alokasi buffer lokal (pastikan pointer non-NULL walau jumlah 0) */
    int local_rows = rows_per_proc[rank];
    int *local_matrix = NULL;
    if (local_rows > 0) {
        local_matrix = malloc(local_rows * COLS * sizeof(int));
    } else {
        local_matrix = malloc(sizeof(int)); // dummy pointer untuk beberapa implementasi MPI
    }

    /* scatter the rows (sebagai blok elemen) */
    MPI_Scatterv(
        matrix, sendcounts, displs_elems, MPI_INT,
        local_matrix, local_rows * COLS, MPI_INT,
        0, MPI_COMM_WORLD
    );
```

```c
    /* hitung jumlah tiap baris lokal */
    int *local_sums = NULL;
    if (local_rows > 0) {
        local_sums = malloc(local_rows * sizeof(int));
        for (int r = 0; r < local_rows; r++) {
            int sum = 0;
            for (int c = 0; c < COLS; c++) {
                sum += local_matrix[r * COLS + c];
            }
            local_sums[r] = sum;
        }
    } else {
        local_sums = malloc(sizeof(int)); // dummy
    }

    /* siapkan recvcounts/displs untuk Gatherv (hanya root butuh) */
    int *recvcounts_rows = NULL;
    int *displs_rows = NULL;
    int *total_sums = NULL;
    if (rank == 0) {
        recvcounts_rows = malloc(size * sizeof(int));
        displs_rows = malloc(size * sizeof(int));
        int off = 0;
        for (int i = 0; i < size; i++) {
            recvcounts_rows[i] = rows_per_proc[i];
            displs_rows[i] = off;
            off += recvcounts_rows[i];
        }
        total_sums = malloc(ROWS * sizeof(int));
    }
```

```
88
89      /* kumpulkan semua jumlah baris di root */
90      MPI_Gatherv(
91          local_sums, local_rows, MPI_INT,
92          total_sums, recvcounts_rows, displs_rows, MPI_INT,
93          0, MPI_COMM_WORLD
94      );
95
96      /* root tampilkan hasil */
97      if (rank == 0) {
98          for (int i = 0; i < ROWS; i++) {
99              printf("Row %3d sum = %d\n", i + 1, total_sums[i]);
100         }
101         /* (opsional) cek grand total */
102         long grand = 0;
103         for (int i = 0; i < ROWS; i++) grand += total_sums[i];
104         printf("Grand total = %ld\n", grand);
105     }
106
```

```
101         /* (opsional) cek grand total */
102         long grand = 0;
103         for (int i = 0; i < ROWS; i++) grand += total_sums[i];
104         printf("Grand total = %ld\n", grand);
105     }
106
107     /* free semua */
108     if (matrix) free(matrix);
109     if (local_matrix) free(local_matrix);
110     if (local_sums) free(local_sums);
111     free(sendcounts);
112     free(displs_elems);
113     free(rows_per_proc);
114     if (recvcounts_rows) free(recvcounts_rows);
115     if (displs_rows) free(displs_rows);
116     if (total_sums) free(total_sums);
117
118     MPI_Finalize();
119     return 0;
120 }
121
```

Hasil:

```
Rank 1: menangani baris global 51 .. 100 (jumlah baris = 50)
Rank 0: menangani baris global 1 .. 50 (jumlah baris = 50)

=== Jumlah tiap baris (1..100) ===
Baris   1: 4843
Baris   2: 4744
Baris   3: 4915
Baris   4: 5223
Baris   5: 4831
Baris   6: 4829
Baris   7: 4772
Baris   8: 5178
Baris   9: 5078
Baris  10: 4993
Baris  11: 5074
Baris  12: 5317
Baris  13: 4732
Baris  14: 4901
Baris  15: 4922
Baris  16: 5037
Baris  17: 5185
Baris  18: 5140
Baris  19: 4998
Baris  20: 4981
Baris  21: 5265
Baris  22: 5096
Baris  23: 4773
```

```
Baris  24: 4643
Baris  25: 5020
Baris  26: 5070
Baris  27: 4916
Baris  28: 5015
Baris  29: 4782
Baris  30: 5240
Baris  31: 5031
Baris  32: 5149
Baris  33: 4965
Baris  34: 4757
Baris  35: 5221
Baris  36: 4972
Baris  37: 4886
Baris  38: 5494
Baris  39: 5095
Baris  40: 4872
Baris  41: 5106
Baris  42: 4946
Baris  43: 5228
Baris  44: 5123
Baris  45: 5255
Baris  46: 4992
Baris  47: 4563
Baris  48: 4716
Baris  49: 4976
Baris  50: 4717
Baris  51: 5061
Baris  52: 5393
```

```
Baris  53: 4982
Baris  54: 5292
Baris  55: 5247
Baris  56: 4493
Baris  57: 5043
Baris  58: 4806
Baris  59: 5127
Baris  60: 4533
Baris  61: 5294
Baris  62: 4851
Baris  63: 4920
Baris  64: 4860
Baris  65: 5138
Baris  66: 4956
Baris  67: 4601
Baris  68: 5075
Baris  69: 4706
Baris  70: 5166
Baris  71: 5037
Baris  72: 5013
Baris  73: 4581
Baris  74: 5156
Baris  75: 4610
Baris  76: 5078
Baris  77: 4936
Baris  78: 4672
Baris  79: 4653
Baris  80: 4969
Baris  81: 5107
```

```
Baris  82: 4587
Baris  83: 4660
Baris  84: 5865
Baris  85: 4633
Baris  86: 5442
Baris  87: 4710
Baris  88: 5023
Baris  89: 4217
Baris  90: 4744
Baris  91: 4966
Baris  92: 5160
Baris  93: 3984
Baris  94: 4766
Baris  95: 4966
Baris  96: 4929
Baris  97: 5193
Baris  98: 4583
Baris  99: 4769
Baris 100: 4920
```

```
Rank 1: menangani baris global 35 .. 67 (jumlah baris = 33)
Rank 0: menangani baris global 1 .. 34 (jumlah baris = 34)
Rank 2: menangani baris global 68 .. 100 (jumlah baris = 33)

=== Jumlah tiap baris (1..100) ===
Baris   1: 5025
Baris   2: 5064
Baris   3: 5118
Baris   4: 5535
Baris   5: 5150
Baris   6: 4598
Baris   7: 4815
Baris   8: 5257
Baris   9: 5194
Baris  10: 5266
Baris  11: 4510
Baris  12: 4681
Baris  13: 5026
Baris  14: 5613
Baris  15: 4841
Baris  16: 4494
Baris  17: 5297
Baris  18: 5026
Baris  19: 5405
Baris  20: 4544
Baris  21: 5223
Baris  22: 5095
Baris  23: 4894
Baris  24: 4504
```

```
Baris  25: 4763
Baris  26: 5006
Baris  27: 4870
Baris  28: 5465
Baris  29: 5331
Baris  30: 5118
Baris  31: 5394
Baris  32: 4718
Baris  33: 5289
Baris  34: 5289
Baris  35: 5119
Baris  36: 5106
Baris  37: 4440
Baris  38: 5093
Baris  39: 4814
Baris  40: 4764
Baris  41: 5735
Baris  42: 4845
Baris  43: 5013
Baris  44: 4702
Baris  45: 4399
Baris  46: 4913
Baris  47: 4314
Baris  48: 4494
Baris  49: 4652
Baris  50: 4712
Baris  51: 4529
Baris  52: 5051
Baris  53: 5006
```

```
Baris  54: 5069
Baris  55: 4929
Baris  56: 5182
Baris  57: 5455
Baris  58: 5196
Baris  59: 4861
Baris  60: 4704
Baris  61: 5241
Baris  62: 4892
Baris  63: 5174
Baris  64: 4928
Baris  65: 4923
Baris  66: 4940
Baris  67: 4517
Baris  68: 5092
Baris  69: 4758
Baris  70: 4684
Baris  71: 4521
Baris  72: 5298
Baris  73: 5222
Baris  74: 4979
Baris  75: 4515
Baris  76: 4606
Baris  77: 4876
Baris  78: 4776
Baris  79: 4565
Baris  80: 5146
Baris  81: 5111
Baris  82: 5026
```

```
Baris  83: 4549
Baris  84: 5218
Baris  85: 4817
Baris  86: 4641
Baris  87: 5423
Baris  88: 5246
Baris  89: 5201
Baris  90: 4570
Baris  91: 4839
Baris  92: 4883
Baris  93: 5230
Baris  94: 4654
Baris  95: 4455
Baris  96: 4963
Baris  97: 4535
Baris  98: 5201
Baris  99: 5582
Baris 100: 5167
```